Lab 3

Zara Waheed

15th Feb 2022

Question 5.8

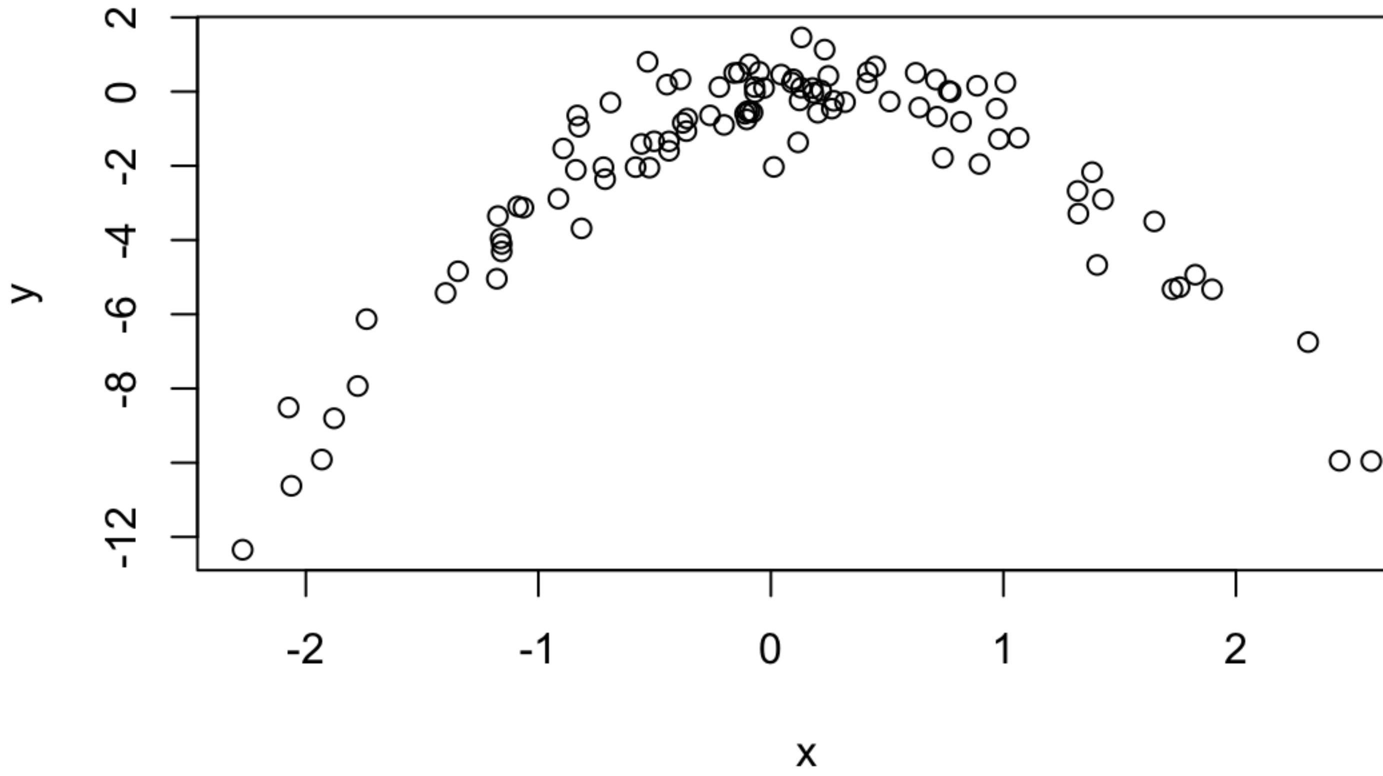
a)

```
set.seed(100)
x = rnorm(100)
y = x - 2*x^2 + rnorm(100)
```

n = 100 p = 2 Y= X − 2X^2 + ϵ

b)

```
plot(x, y)
```



The equation is quadratic as can be seen from the plot and the approximate ranges of x and y are -2 to 2 and -8 to 2, respectively.

c)

```
df = data.frame(x, y)
set.seed(100)

# i) Y = B0 + B1*X + ϵ
fit1 = glm(y ~ x)
cv.glm(df, fit1)$delta
## [1] 9.060636 9.056661
# ii) Y = B0 + B1*X + B2*X^2 + ϵ
fit2 = glm(y ~ poly(x, 2))
cv.glm(df, fit2)$delta
## [1] 0.6511909 0.6509495
# iii) Y = B0 + B1*X + B2*X^2 + B3*X^3 + ϵ
fit3 = glm(y ~ poly(x, 3))
cv.glm(df, fit3)$delta
## [1] 0.6665339 0.6661944
# iv) Y = B0 + B1*X + B2*X^2 + B3*X^3 + B4*X^4 + ϵ
fit4 = glm(y ~ poly(x, 4))
cv.glm(df, fit4)$delta
## [1] 0.6671261 0.6667107
```

d)

```
set.seed(100)

# i) Y = B0 + B1*X + ϵ
fit5 = glm(y ~ x)
cv.glm(df, fit5)$delta
## [1] 9.060636 9.056661
# ii) Y = B0 + B1*X + B2*X^2 + ϵ
fit6 = glm(y ~ poly(x, 2))
cv.glm(df, fit6)$delta
## [1] 0.6511909 0.6509495
# iii) Y = B0 + B1*X + B2*X^2 + B3*X^3 + ϵ
fit7 = glm(y ~ poly(x, 3))
cv.glm(df, fit7)$delta
## [1] 0.6665339 0.6661944
# iv) Y = B0 + B1*X + B2*X^2 + B3*X^3 + B4*X^4 + ϵ
fit8 = glm(y ~ poly(x, 4))
cv.glm(df, fit8)$delta
## [1] 0.6671261 0.6667107
```

The results are exactly the same as part c).

e)

Equation ii) had the lowest error rate, which could be because it is in quadratic form and is similar to the y equation.

f)

```
summary(fit1)
##
## Call:
## glm(formula = y ~ x)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.313  -1.212   1.125   1.968   3.439
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.0504     0.2908  -7.051 2.52e-10 ***
## x             0.5450     0.2863   1.903   0.0599 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 8.456659)
##
##     Null deviance: 859.39  on 99  degrees of freedom
## Residual deviance: 828.75  on 98  degrees of freedom
## AIC: 501.26
##
## Number of Fisher Scoring iterations: 2
```

Yes these results agree with the cross validation results.

Question 6.9

a)

```
data(College)
set.seed(1)
df1 <- sample(1:dim(College)[1], dim(College)[1] / 2)
df2 <- -df1
train <- College[df1, ]
test <- College[df2, ]
```

b)

```
fit9b <- lm(Apps ~ ., data = train)
pred.lm <- predict(fit9b, test)
mean((pred.lm - test$Apps)^2)
## [1] 1135758
```

c)

```
set.seed(100)

train.mx <- model.matrix(Apps ~., data = train[ ,-1])
test.mx <- model.matrix(Apps ~., data = test[ ,-1])
cv.ridge <- cv.glmnet(train.mx, train$Apps, alpha = 0)
lambda.ridge <- cv.ridge$lambda.min


pred.ridge <- predict(cv.ridge, s = lambda.ridge, newx = test.mx)
mean((pred.ridge - test$Apps) ^2)
## [1] 1007688
lambda.ridge
## [1] 405.8404
```

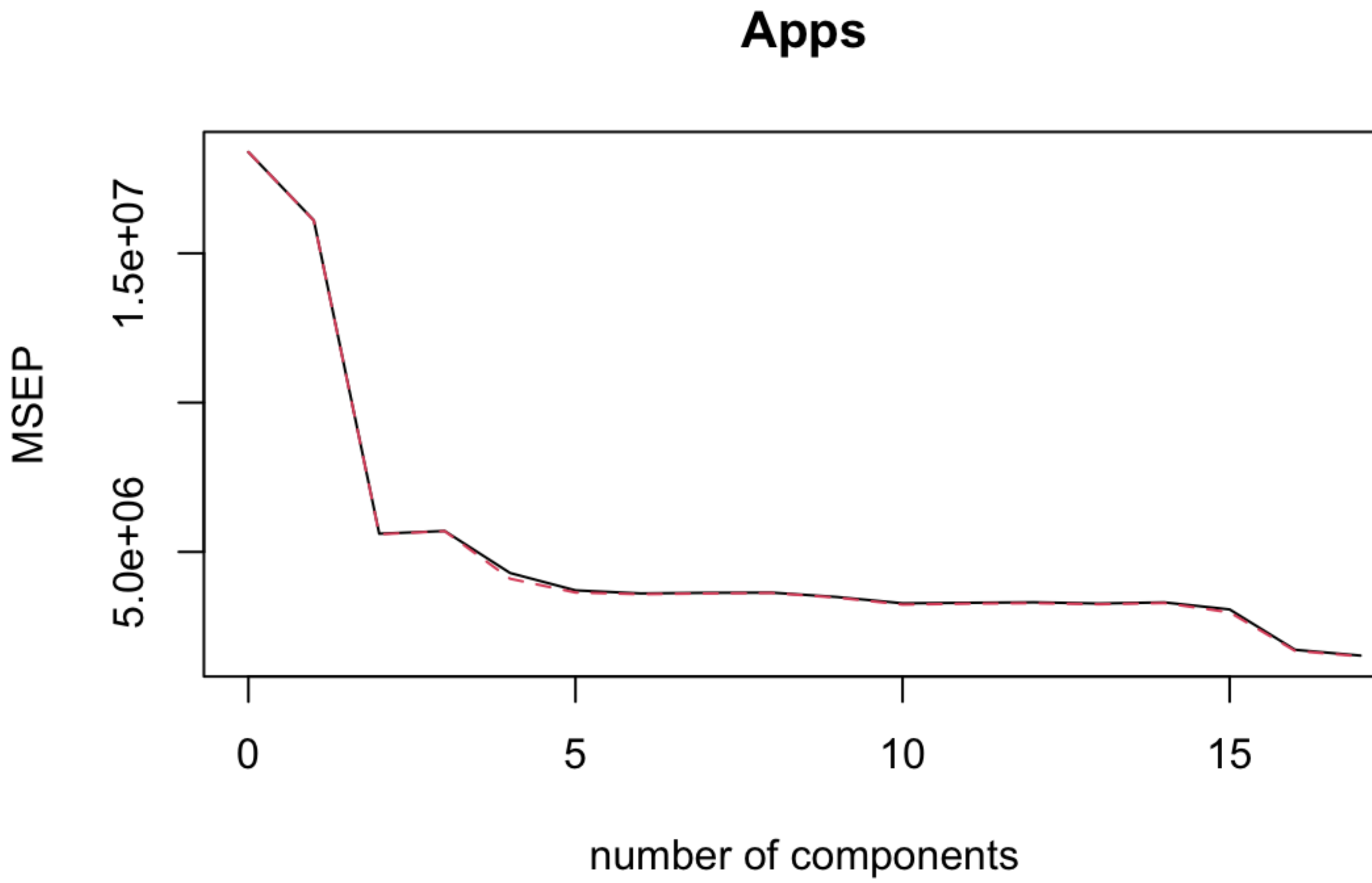
d)

```
set.seed(1)
cv.lasso <- cv.glmnet(train.mx, train$Apps, alpha = 1)
lambda.lasso <- cv.lasso$lambda.min
lambda.lasso
## [1] 2.165848
pred.lasso <- predict(cv.lasso, s = lambda.lasso, newx = test.mx)
mean((pred.lasso - test$Apps) ^2)
## [1] 1140473
```

e)

```
pcr.fit <- pcr(Apps ~ ., data = train, scale = TRUE, validation = "CV")
summary(pcr.fit)
## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: svdpc
```

```
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4288     4013     2368     2388     2072     1926     1900
## adjCV         4288     4012     2364     2386     2025     1907     1893
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1905     1907     1868      1811      1815      1820      1809
## adjCV     1899     1903     1862      1799      1807      1812      1801
##        14 comps  15 comps  16 comps  17 comps
## CV         1819      1753      1312      1236
## adjCV      1813      1725      1298      1225
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8
comps
## X       32.20    57.78    65.31    70.99    76.37    81.27     84.8
87.85
## Apps    13.44    70.93    71.07    79.87    81.15    82.25     82.3
82.33
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15
comps
## X       90.62     92.91     94.98     96.74     97.79     98.72
99.42
## Apps    83.38     84.76     84.80     84.84     85.11     85.14
90.55
##       16 comps  17 comps
## X        99.88    100.00
## Apps     93.42     93.89
validationplot(pcr.fit, val.type = "MSEP")
```

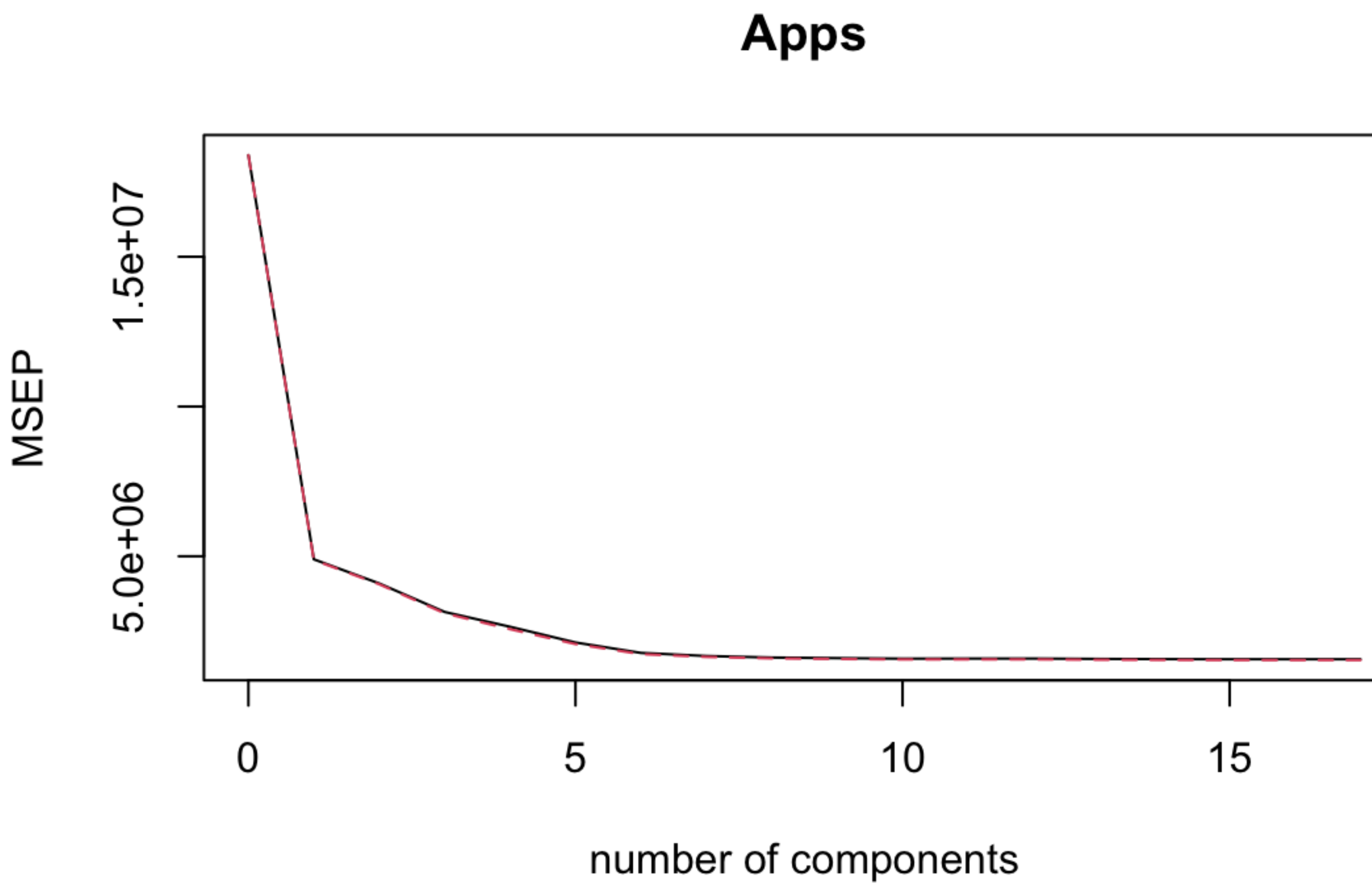



```
pred.pcr <- predict(pcr.fit, test.mx, ncomp = 5)
mean((pred.pcr - test$Apps)^2)
## [1] 1963819
```

f)

```
set.seed(100)

pls.fit <- plsr(Apps ~ ., data = train, scale = TRUE, validation = "CV")
summary(pls.fit)
## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4288     2213     2023     1772     1627     1457     1332
## adjCV         4288     2208     2016     1759     1601     1438     1316
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1293     1272     1264      1258      1259      1259      1254
## adjCV     1280     1261     1253      1247      1247      1247      1243
##        14 comps  15 comps  16 comps  17 comps
```

```
## CV          1252       1250       1250       1250
## adjCV       1240       1239       1238       1238
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8
comps
## X        27.21    50.73    63.06    65.52    70.20    74.20    78.62
80.81
## Apps     75.39    81.24    86.97    91.14    92.62    93.43    93.56
93.68
##        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15
comps
## X        83.29     87.17     89.15     91.37     92.58     94.42
96.98
## Apps     93.76     93.79     93.83     93.86     93.88     93.89
93.89
##        16 comps  17 comps
## X         98.78    100.00
## Apps      93.89     93.89
validationplot(pls.fit, val.type = "MSEP")
```




```
pred.pls <- predict(pls.fit, test.mx, ncomp = 10)
mean((pred.pls - test$Apps)^2)
## [1] 1181808
```

g)

```
# Calculate R^2 for all models

test.avg <- mean(test$Apps)
lm <- 1- mean((pred.lm - test$Apps)^2) / mean((test.avg - test$Apps)^2)
ridge <- 1- mean((pred.ridge - test$Apps)^2) / mean((test.avg -
test$Apps)^2)
lasso <- 1- mean((pred.lasso - test$Apps)^2) / mean((test.avg -
test$Apps)^2)
pcr <- 1- mean((pred.pcr - test$Apps)^2) / mean((test.avg - test$Apps)^2)
pls <- 1- mean((pred.pls - test$Apps)^2) / mean((test.avg - test$Apps)^2)

lm
## [1] 0.9015413
ridge
## [1] 0.9126437
lasso
## [1] 0.9011326
pcr
## [1] 0.8297569
pls
## [1] 0.8975493
```

All of the models are fairly accurate except PCR

Question 6.10

a)

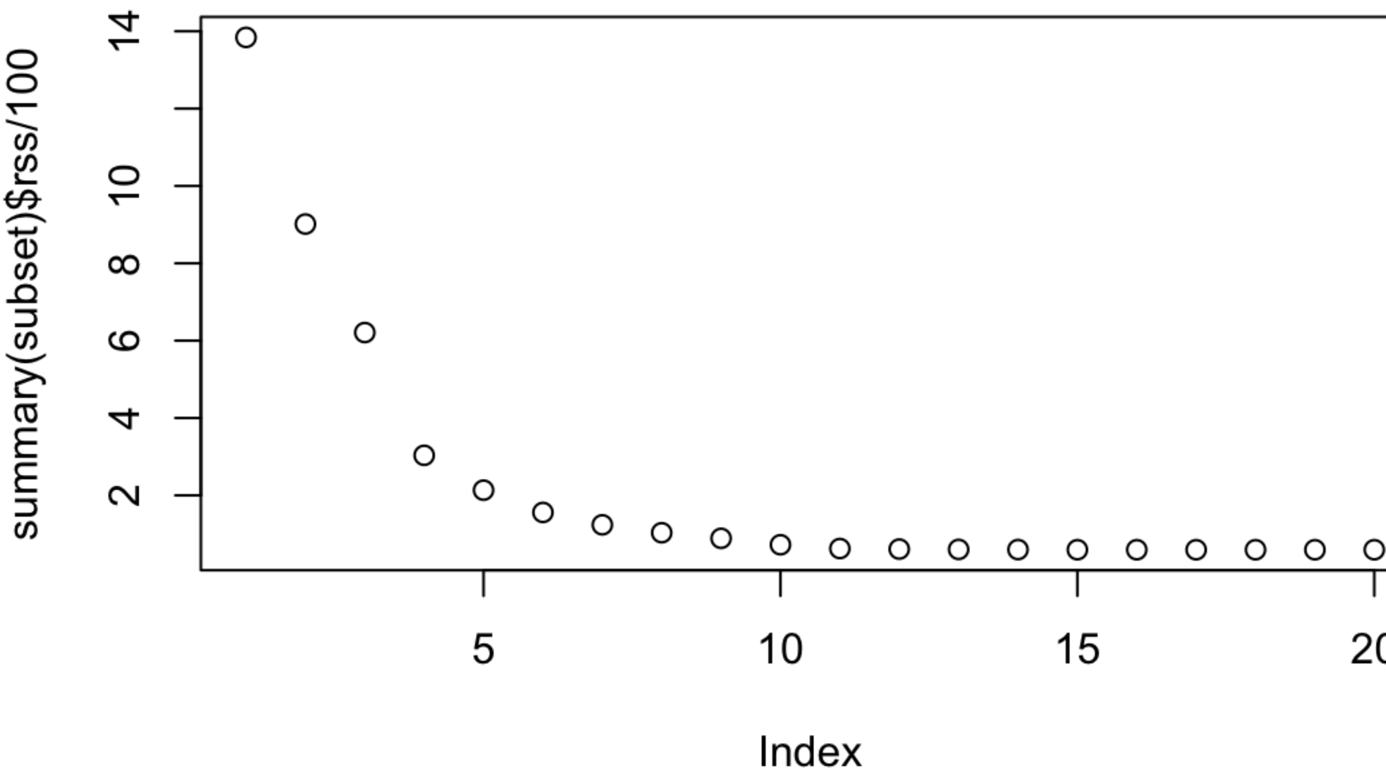
```
set.seed(100)

p = 20
n = 1000
x = matrix(rnorm(n*p),n,p)
B = rnorm(p)
B[c(2,3,8,9,10,15,20)] = 0
e = rnorm(n)
y = x %*% B + e
```

b)

```
data <- data.frame(x, y)
#train = sample(seq(1000),100,replace=F)
train <- data[1:100,]
test <- data[101:1000,]
```

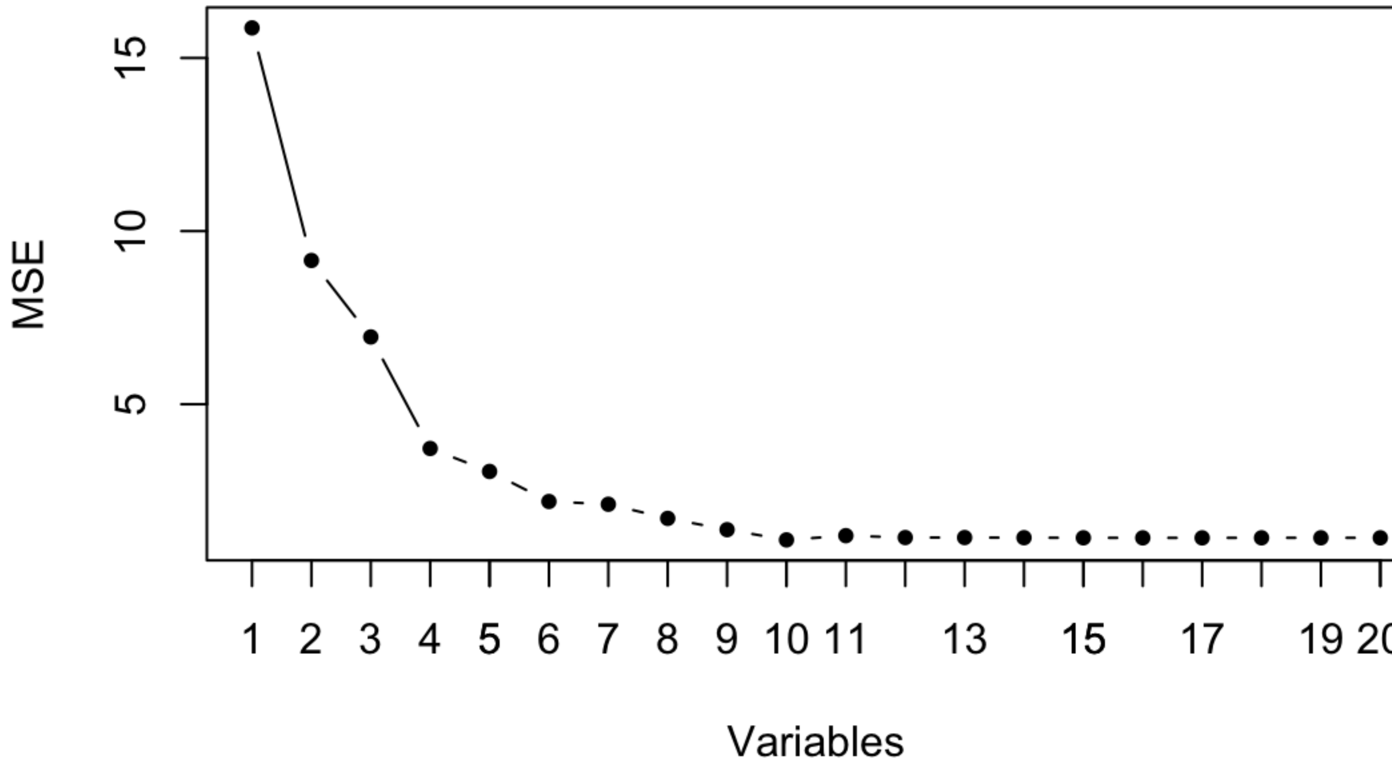
c)

```
subset = regsubsets(y~.,train, nvmax=p)
plot(summary(subset)$rss/100)
```



d)

```
test.mx <- model.matrix(y ~., test, nvmax = 20)
errors <- rep(NA, 20)
for (i in 1:20) {
  coef <- coef(subset, id = i)
  pred <- test.mx[, names(coef)] %*%coef
  errors[i] <- mean((pred - test[,21])^2)
}
plot(errors, xlab = "Variables", ylab = "MSE", type = "b", pch = 20)
axis(1, at = seq(1, 20, 1))
```



e)

```
which.min(errors)
## [1] 10
```

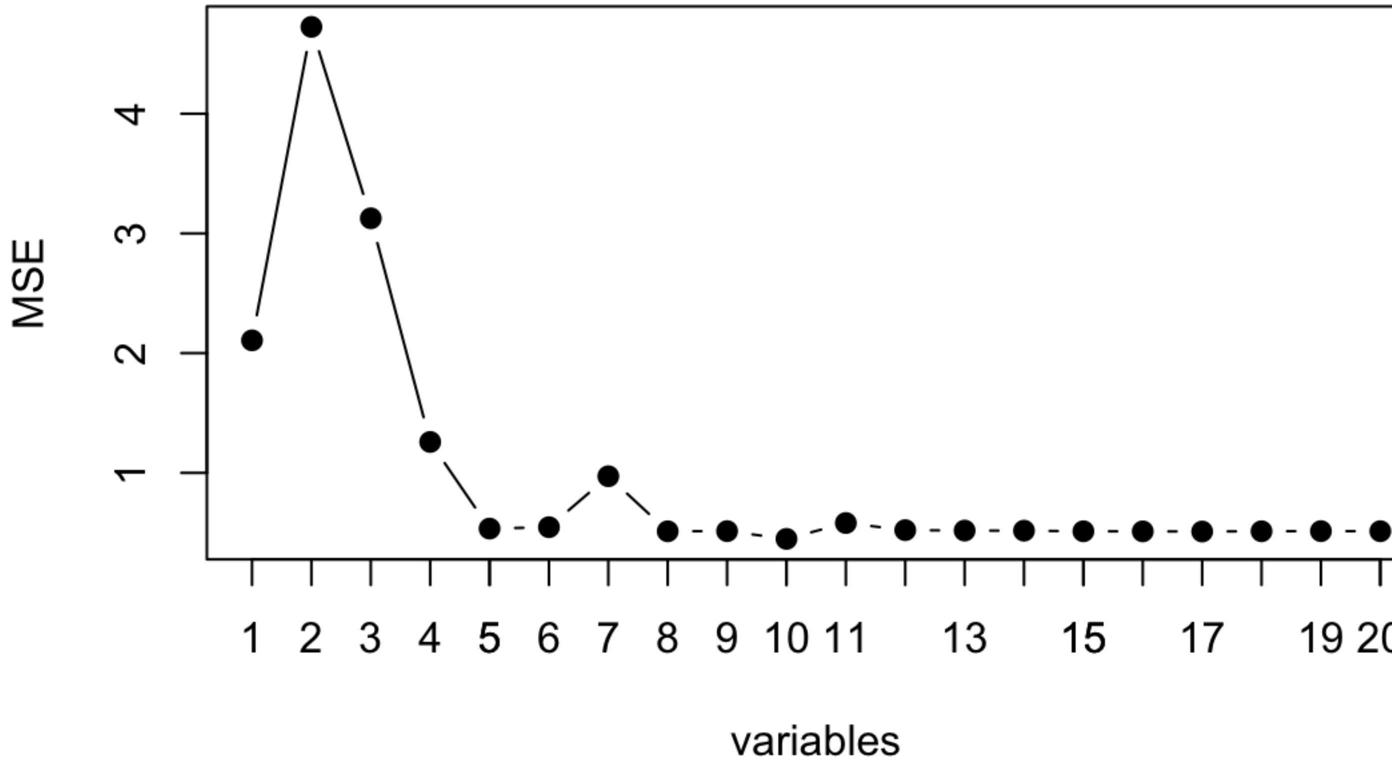
f)

```
coef(subset, which.min(errors))
## (Intercept)          X1          X6         X11         X12         X13
## -0.06800658 -0.45731970  1.84058033  0.75763718  0.44049339 -2.48741657
##         X14         X16         X17         X18         X19
## -0.49962102  1.62658317  0.66224111  0.94966314 -3.10397368
```

g)

```
errors <- rep(NA, 20)
x_colname <- colnames(x, do.NULL = FALSE, prefix = "X")
for (i in 1:20) {
  coeff <- coef(subset, id = i)
```

```
  errors[i] <- sqrt(sum((B[x_colname %in% names(coeff)] -
coeff[names(coeff) %in% x_colname])^2) + sum(B[!(x_colname %in%
names(coeff))])^2)
}
plot(errors, xlab = "variables", ylab = "MSE", type = "b", pch = 19)
axis(1, at = seq(1, 20, 1))
```



The plot shows a drop in the coefficient error.